Modul 223 Projectarbeit:

Forum :Gringo

Merlin luedi, Lorik Murtesi, Manuel Vacirca

25.01.2025

Inhalt

[1. Einleitung und Anforderungsanalyse (User Stories) 3](#_Toc188704544)

[2. Sicherheitskonzept 3](#_Toc188704545)

[3. Arbeitsplanung 3](#_Toc188704546)

[4. Test-Konzept 3](#_Toc188704547)

[5. Beschreibung der eingesetzten Frameworks 4](#_Toc188704548)

[6. Beschreibung der Abläufe beim Login 4](#_Toc188704549)

[7. Testprotokoll 4](#_Toc188704550)

[8. Arbeitsjournal 4](#_Toc188704551)

# 1. Einleitung und Anforderungsanalyse (User Stories)

**2. Einleitung und Anforderungsanalyse**

**Einleitung**

Das Ziel dieses Projekts ist die Entwicklung einer Multiuser-Webanwendung, die als interaktives Forum dient. Benutzer können sich registrieren, anmelden und Beiträge zu verschiedenen Foren verfassen. Dabei werden moderne Technologien wie **Spring Boot** für das Backend und **React** für das Frontend eingesetzt.  
Das Forum ermöglicht es Benutzern, Ideen auszutauschen, Fragen zu stellen und mit anderen Mitgliedern zu kommunizieren. Besondere Aufmerksamkeit wird auf die Sicherheit gelegt, um sensible Benutzerdaten zu schützen und einen reibungslosen Ablauf zu gewährleisten.

Die Webanwendung soll den gesamten Prozess einer Multiuser-Interaktion abbilden, von der Registrierung und Authentifizierung bis hin zur Erstellung und Verwaltung von Foren und Beiträgen. Die Anforderungen werden durch klare User Stories und eine strukturierte Planung abgedeckt.

**Anforderungsanalyse**

Für die erfolgreiche Entwicklung des Forums wurden die folgenden Anforderungen identifiziert:

**Funktionale Anforderungen**:

1. **Registrierung und Login**
   * Benutzer können sich mit einem Benutzernamen und einem Passwort registrieren.
   * Ein sicherer Login-Prozess mit JWT-Token wird implementiert.
2. **Forenverwaltung**
   * Benutzer können neue Foren erstellen, bestehende Foren anzeigen und in Foren Beiträge verfassen.
   * Beiträge werden einem spezifischen Forum zugeordnet.
3. **Benutzerfreundlichkeit**
   * Intuitive Benutzeroberfläche, die es Benutzern ermöglicht, leicht zwischen Foren und Beiträgen zu navigieren.
4. **Rechtemanagement**
   * Nur angemeldete Benutzer können Foren erstellen und Beiträge verfassen.
   * Öffentlich zugängliche Foren und Beiträge für nicht angemeldete Benutzer.

**Nicht-funktionale Anforderungen**:

1. **Sicherheit**
   * Verwendung von HTTPS und JWT zur sicheren Kommunikation.
   * Sichere Speicherung von Passwörtern mit Hashing.
2. **Performance**
   * Die Anwendung soll schnell reagieren und auch bei einer hohen Anzahl von Benutzern stabil laufen.
3. **Wartbarkeit**
   * Klar strukturierter und dokumentierter Code für zukünftige Erweiterungen.

**User Stories**

Die Anforderungen werden durch die folgenden User Stories abgebildet:

1. **Registrierung und Login**
   * Als Benutzer möchte ich mich registrieren können, damit ich Zugriff auf die Funktionen der Plattform habe.
   * Als Benutzer möchte ich mich sicher anmelden können, damit ich meine Aktivitäten personalisieren kann.
2. **Forenverwaltung**
   * Als Benutzer möchte ich ein neues Forum erstellen können, um ein spezifisches Thema zu diskutieren.
   * Als Benutzer möchte ich eine Liste aller verfügbaren Foren sehen, um Themen von Interesse zu finden.
   * Als Benutzer möchte ich ein Forum auswählen können, um alle Beiträge zu diesem Thema anzuzeigen.
3. **Beitragsverwaltung**
   * Als Benutzer möchte ich Beiträge in einem Forum verfassen, um meine Gedanken zu teilen.
   * Als Benutzer möchte ich eine Liste aller Beiträge in einem Forum sehen, um Diskussionen zu verfolgen.
4. **Sicherheit und Autorisierung**
   * Als Benutzer möchte ich, dass meine persönlichen Daten sicher gespeichert werden, um Datenschutz zu gewährleisten.
   * Als Gast möchte ich öffentliche Foren durchsuchen können, ohne mich anmelden zu müssen.

# 2. Sicherheitskonzept

Die Sicherheit der Multiuser-Webanwendung ist ein zentraler Bestandteil des Projekts, um die Integrität und Vertraulichkeit der Benutzerdaten zu gewährleisten und unbefugten Zugriff zu verhindern. Im Rahmen dieses Projekts wurden mehrere Sicherheitsmaßnahmen implementiert, die in verschiedenen Bereichen der Anwendung greifen.

**1. Authentifizierung und Autorisierung**

1. **JWT-Token (JSON Web Token)**
   * Nach erfolgreicher Anmeldung erhält der Benutzer ein JWT-Token, das bei jeder nachfolgenden Anfrage an geschützte API-Endpunkte mitgesendet wird.
   * Das Token enthält die Identität des Benutzers und hat eine begrenzte Gültigkeitsdauer, um Missbrauch zu minimieren.
2. **Sicherer Login-Prozess**
   * Benutzeranmeldedaten werden mit sicheren Algorithmen gehasht (z. B. BCrypt) und in der Datenbank gespeichert.
   * Übermittelte Passwörter werden niemals im Klartext gespeichert oder übertragen.

**2. Datensicherheit**

1. **Verschlüsselte Kommunikation**
   * Die Anwendung verwendet HTTPS, um die Daten zwischen Frontend und Backend zu verschlüsseln und vor Abhören oder Manipulation zu schützen.
2. **Datenvalidierung und -sanitierung**
   * Eingaben von Benutzern werden sowohl im Frontend als auch im Backend validiert, um SQL-Injection und Cross-Site Scripting (XSS) zu verhindern.
   * Spezifische Parameter (z. B. Foren-ID, Benutzername) werden vor der Verarbeitung überprüft, um Manipulationen auszuschließen.
3. **Fehlerbehandlung**
   * Fehlermeldungen sind generisch gehalten, um potenziellen Angreifern keine zusätzlichen Informationen zu liefern.
   * Beispiel: Statt "Benutzername existiert nicht" lautet die Fehlermeldung "Ungültige Anmeldeinformationen".

**3. Zugriffskontrolle**

1. **Backend-Schutz durch Filter**
   * Die Anwendung verwendet einen JWT-Filter, um sicherzustellen, dass nur autorisierte Anfragen an geschützte API-Endpunkte zugelassen werden.
   * Anfragen ohne gültiges JWT-Token erhalten eine HTTP-403-Fehlermeldung (Forbidden).
2. **Gastzugriff auf öffentliche Inhalte**
   * Öffentliche API-Endpunkte (z. B. zum Abrufen der Liste aller Foren) sind für Gäste zugänglich.
   * Aktionen wie das Erstellen eines Forums oder Beitrags sind auf authentifizierte Benutzer beschränkt.

**4. Schutz vor gängigen Sicherheitsrisiken**

1. **Cross-Site Scripting (XSS)**
   * Alle Benutzereingaben werden vor der Anzeige auf potenziell gefährliche Inhalte überprüft und entschärft.
   * Beispiel: HTML- oder JavaScript-Code, der als Beitrag erstellt wird, wird sicher gerendert, um XSS-Angriffe zu verhindern.
2. **SQL-Injection**
   * Alle Datenbankzugriffe erfolgen über JPA (Java Persistence API), wodurch SQL-Injection-Angriffe verhindert werden.
3. **CSRF (Cross-Site Request Forgery)**
   * Schutzmechanismen wie spezifische HTTP-Header werden verwendet, um sicherzustellen, dass Anfragen nur von vertrauenswürdigen Quellen stammen.

**5. Sicherheitsmaßnahmen für sensible Daten**

1. **Passwort-Hashing**
   * Benutzerpasswörter werden mit einem starken Algorithmus (BCrypt) gehasht und in der Datenbank gespeichert.
2. **Token-Invalidierung**
   * JWT-Tokens haben eine zeitlich begrenzte Gültigkeit.
   * Bei Logout oder Änderung der Benutzerdaten kann das Token invalidiert werden.
3. **Sicherung der Backend-Konfiguration**
   * Sensible Daten wie Datenbankpasswörter oder JWT-Schlüssel werden in einer sicheren Konfigurationsdatei oder Umgebungsvariablen gespeichert.

**6. Protokollierung und Überwachung**

1. **Logging von sicherheitsrelevanten Ereignissen**
   * Alle sicherheitsrelevanten Ereignisse (z. B. fehlgeschlagene Login-Versuche, ungültige Token) werden protokolliert.
   * Diese Logs helfen bei der Fehlerbehebung und bei der Erkennung von Sicherheitsvorfällen.
2. **Überwachung von API-Endpunkten**
   * Alle API-Endpunkte werden regelmäßig auf Missbrauch überprüft, z. B. durch Rate Limiting oder IP-Blockierung bei verdächtigen Anfragen.

**Zusammenfassung der Sicherheitsmaßnahmen**

* Sichere Authentifizierung durch JWT und Passwort-Hashing.
* Verschlüsselte Datenübertragung mit HTTPS.
* Validierung und Überprüfung von Benutzereingaben.
* Schutz vor gängigen Angriffen wie XSS und SQL-Injection.
* Protokollierung sicherheitsrelevanter Ereignisse.

# 3. Arbeitsplanung

Die Arbeitsplanung ist in sinnvolle Arbeitspakete unterteilt, die eine strukturierte und effiziente Entwicklung der Multiuser-Webanwendung ermöglichen. Die Schätzungen basieren auf der Komplexität und dem Umfang der Aufgaben.

**1. Analyse und Konzeption**

* **Ziel**: Anforderungsanalyse und Erstellung eines groben Konzepts für die Anwendung.
* **Aufgaben**:
  + Sammlung und Definition der User Stories.
  + Erstellung des Sicherheitskonzepts.
  + Skizze der Datenbankstruktur.
* **Geschätzter Zeitaufwand**: 4 Stunden.

**2. Einrichtung der Entwicklungsumgebung**

* **Ziel**: Sicherstellung, dass alle Tools und Frameworks korrekt eingerichtet sind.
* **Aufgaben**:
  + Installation von Spring Boot und React.
  + Einrichtung des Git-Repositories.
  + Konfiguration von Datenbank und API-Endpunkten.
* **Geschätzter Zeitaufwand**: 2 Stunden.

**3. Backend-Entwicklung**

* **Ziel**: Entwicklung der serverseitigen Logik und Datenbankinteraktionen.
* **Aufgaben**:
  + Implementierung der API-Endpunkte (z. B. Login, Registrierung, Foren- und Post-Verwaltung).
  + Einrichtung der Sicherheitsmechanismen (z. B. JWT-Authentifizierung, Passwort-Hashing).
  + Validierung und Fehlerbehandlung.
* **Geschätzter Zeitaufwand**: 10 Stunden.

**4. Frontend-Entwicklung**

* **Ziel**: Entwicklung der Benutzeroberfläche mit React.
* **Aufgaben**:
  + Erstellung der Seitenstruktur (z. B. Login, Registrierung, Forum-Liste, Post-Liste).
  + Integration der API-Endpunkte mit Axios oder Fetch.
  + Implementierung von Formulareingaben und Validierung.
  + Routing und Navigation zwischen Seiten.
* **Geschätzter Zeitaufwand**: 12 Stunden.

**5. Verknüpfung von Frontend und Backend**

* **Ziel**: Sicherstellen, dass die Benutzeroberfläche korrekt mit dem Backend kommuniziert.
* **Aufgaben**:
  + Implementierung von Login- und Registrierungsabläufen.
  + Anzeigen und Verwalten von Foren und Posts.
  + Fehlerbehandlung bei der API-Kommunikation.
* **Geschätzter Zeitaufwand**: 6 Stunden.

**6. Tests und Debugging**

* **Ziel**: Sicherstellen, dass die Anwendung stabil und fehlerfrei ist.
* **Aufgaben**:
  + Unit-Tests für das Backend (Spring Boot).
  + Integrationstests für die API-Endpunkte.
  + Manuelle Tests der Frontend-Funktionalität.
  + Behebung von Bugs.
* **Geschätzter Zeitaufwand**: 8 Stunden.

**7. Dokumentation**

* **Ziel**: Erstellung der Projektdokumentation gemäß den Anforderungen.
* **Aufgaben**:
  + Schreiben der Einleitung und Anforderungsanalyse.
  + Beschreibung der Sicherheitskonzepte, Arbeitsplanung und Frameworks.
  + Erstellung des Testprotokolls und Arbeitsjournals.
* **Geschätzter Zeitaufwand**: 6 Stunden.

**8. Präsentationsvorbereitung**

* **Ziel**: Vorbereitung der Projektvorstellung und Fachgespräch.
* **Aufgaben**:
  + Erstellung einer Präsentation.
  + Probelauf der Demonstration der Anwendung.
* **Geschätzter Zeitaufwand**: 2 Stunden.

**Zeitplanübersicht**

| **Arbeitspaket** | **Zeitaufwand** |
| --- | --- |
| Analyse und Konzeption | 4 Stunden |
| Einrichtung der Entwicklungsumgebung | 2 Stunden |
| Backend-Entwicklung | 10 Stunden |
| Frontend-Entwicklung | 12 Stunden |
| Verknüpfung von Frontend und Backend | 6 Stunden |
| Tests und Debugging | 8 Stunden |
| Dokumentation | 6 Stunden |
| Präsentationsvorbereitung | 2 Stunden |
| **Gesamtzeit** | **50 Stunden** |

**Priorisierung**

* **Hoch**: Authentifizierung, Forums- und Post-Funktionalität, Sicherheit.
* **Mittel**: Benutzerfreundlichkeit, Design.
* **Niedrig**: Zusätzliche Features wie Profilerstellung.

# 4. Test-Konzept

Arten von Tests (z. B. Unit-Tests, Integrationstests, End-to-End-Tests)

Testfälle und erwartete Ergebnisse

# 5. Beschreibung der eingesetzten Frameworks

In diesem Projekt wurden zwei Haupt-Frameworks verwendet: **Spring Boot** für das Backend und **React** für das Frontend. Diese Frameworks wurden aufgrund ihrer Leistungsfähigkeit, Flexibilität und Community-Unterstützung ausgewählt.

**Spring Boot (Backend)**

Spring Boot ist ein Java-Framework, das die Erstellung von Microservices und Webanwendungen vereinfacht. Es basiert auf dem Spring Framework und bietet viele vorgefertigte Funktionen für die schnelle Entwicklung moderner Anwendungen.

* **Einsatzbereich im Projekt**:
  + Entwicklung der RESTful-API.
  + Verwaltung der Datenbankoperationen mit JPA (Java Persistence API).
  + Implementierung der Authentifizierung und Autorisierung mithilfe von JWT (JSON Web Tokens).
  + Fehlerbehandlung und Validierung von Benutzereingaben.
* **Vorteile**:
  + **Schnelle Entwicklung**: Durch automatische Konfiguration und Starter-Pakete werden viele Aufgaben vereinfacht.
  + **Flexibilität**: Unterstützung für verschiedene Datenbanken und Frameworks.
  + **Sicherheit**: Einfache Integration von Sicherheitsfeatures wie JWT.
  + **Community-Unterstützung**: Umfangreiche Dokumentation und zahlreiche Bibliotheken.

**React (Frontend)**

React ist eine JavaScript-Bibliothek zur Erstellung von interaktiven Benutzeroberflächen. Sie wurde von Facebook entwickelt und ermöglicht die Entwicklung von komponentenbasierten Single-Page-Anwendungen (SPA).

* **Einsatzbereich im Projekt**:
  + Erstellung von UI-Komponenten für Login, Registrierung, Forum-Liste und Post-Ansicht.
  + Implementierung von Routing und Navigation mithilfe von react-router-dom.
  + Integration der Backend-API mit fetch oder Axios.
  + Verwaltung des Anwendungszustands mit React-Hooks (z. B. useState, useEffect).
* **Vorteile**:
  + **Komponentenbasiert**: Ermöglicht die Wiederverwendung von Code und eine strukturierte Entwicklung.
  + **Effizient**: Dank des Virtual DOM werden UI-Updates schnell und effizient ausgeführt.
  + **Flexibilität**: Kann mit einer Vielzahl von Tools und Bibliotheken kombiniert werden.
  + **Community-Unterstützung**: Umfangreiche Dokumentation, zahlreiche Tutorials und eine aktive Entwickler-Community.

**Zusammenspiel der Frameworks**

* **Spring Boot** stellt die RESTful-API bereit, die vom Frontend konsumiert wird. Die API liefert Daten zu Foren und Posts und ermöglicht die Authentifizierung der Benutzer.
* **React** nutzt die API, um dynamische Inhalte anzuzeigen und Benutzereingaben an das Backend zu senden. Die Kommunikation erfolgt über HTTP-Anfragen.

**Weitere Tools und Technologien**

* **PostgreSQL**: Datenbank für die Speicherung von Benutzern, Foren und Posts.
* **JWT (JSON Web Tokens)**: Wird zur Authentifizierung und Autorisierung verwendet, um sichere API-Aufrufe zu gewährleisten.
* **Git**: Versionskontrollsystem zur Zusammenarbeit und Verfolgung von Änderungen.
* **Postman**: Tool zur API-Testung während der Entwicklung.
* **CSS**: Zur Gestaltung und Verbesserung der Benutzeroberfläche.

# 6. Beschreibung der Abläufe beim Login

Der Login-Prozess in der Anwendung wurde so gestaltet, dass er sowohl sicher als auch benutzerfreundlich ist. Nachfolgend wird der Ablauf beim Login beschrieben, beginnend von der Benutzereingabe bis zur Authentifizierung und Autorisierung.

**1. Benutzeranforderung: Eingabe der Anmeldedaten**

* Der Benutzer gibt seinen Benutzernamen und sein Passwort in die bereitgestellten Eingabefelder auf der Login-Seite ein.
* Nach dem Klick auf den Login-Button wird eine Anfrage an das Backend gesendet.

**2. Frontend-Login-Logik**

* **Validierung im Frontend**: Vor dem Absenden der Daten wird geprüft, ob die Eingaben (Benutzername und Passwort) vollständig und im richtigen Format vorliegen.
* **API-Aufruf**: Die Login-Daten werden über einen HTTP-POST-Aufruf an die /api/auth/login-Route im Backend gesendet. Die Daten werden im JSON-Format übermittelt:
* {
* "username": "testuser",
* "password": "testpassword"
* }

**3. Backend-Authentifizierung**

* **Überprüfung der Anmeldedaten**:
  + Das Backend empfängt die Anfrage und sucht in der Datenbank nach einem Benutzer mit dem angegebenen Benutzernamen.
  + Das eingegebene Passwort wird mit dem gespeicherten, gehashten Passwort verglichen (mittels bcrypt).
* **Erstellung eines JWT-Tokens**:
  + Wenn die Anmeldedaten korrekt sind, generiert das Backend einen JWT (JSON Web Token), der Informationen über den Benutzer (z. B. username) enthält.
  + Das Token wird mit einem geheimen Schlüssel signiert und hat eine begrenzte Gültigkeitsdauer (z. B. 10 Stunden).
* **Antwort des Backends**:
  + Das Backend sendet eine Antwort mit dem generierten Token an das Frontend:
  + {
  + "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  + }
  + Falls die Anmeldedaten ungültig sind, wird eine Fehlermeldung zurückgegeben.

**4. Token-Speicherung im Frontend**

* Das Frontend speichert das erhaltene Token sicher in localStorage oder sessionStorage, damit es für zukünftige API-Aufrufe verwendet werden kann:
* localStorage.setItem("token", token);
* Das Token wird für die Benutzeridentifikation und Zugriffskontrolle genutzt.

**5. Autorisierung bei zukünftigen Anfragen**

* Bei jedem geschützten API-Aufruf (z. B. Erstellen eines Forums oder Posts) wird das Token im Autorisation-Header der Anfrage mitgeschickt:
* Authorization: Bearer <Token>
* Das Backend überprüft das Token:
  + Gültigkeit des Tokens (Signatur und Ablaufdatum).
  + Enthaltene Benutzerinformationen.

**6. Erfolg und Weiterleitung**

* Nach erfolgreichem Login:
  + Der Benutzer wird zur Hauptseite (z. B. der Forumsliste) weitergeleitet.
  + Die Navigation ist entsprechend des authentifizierten Status freigeschaltet.
* Bei fehlgeschlagenem Login:
  + Eine Fehlermeldung wird angezeigt (z. B. "Ungültige Anmeldedaten").

**7. Sicherheitsmaßnahmen**

* **Hashing des Passworts**: Passwörter werden in der Datenbank nur in gehashter Form gespeichert.
* **JWT-Expiration**: Tokens haben eine begrenzte Gültigkeitsdauer, um Missbrauch zu verhindern.
* **HTTPS**: Der Datenverkehr wird verschlüsselt, um Man-in-the-Middle-Angriffe zu verhindern.
* **Logout**:
  + Das Token wird im Frontend gelöscht (localStorage.removeItem("token")).
  + Optional: Das Token kann im Backend ungültig gemacht werden (z. B. durch Blacklisting).

Dieser Login-Prozess gewährleistet die Sicherheit der Benutzerkonten und ermöglicht gleichzeitig eine nahtlose Integration in die Multiuser-Webanwendung.

# 7. Testprotokoll

Tabelle mit durchgeführten Tests, Ergebnissen und Anmerkungen

Beispiel: Testname, Tester, Datum, Status

# 8. Arbeitsjournal

Einträge für jeden Block (z. B. was wurde gemacht, welche Probleme traten auf, wie wurden sie gelöst)

Anhang

Screenshots der Anwendung

Codeausschnitte (falls nötig)

Links zu verwendeten Tools oder Dokumentationen